



uClinux NEEK BSP User Guide

NEEK BSP Version: 2.0.0.0

Document Version: 1.5

Date: September 2008

System Level Solutions, Inc, (USA)

14100 Murphy Avenue,
San Martin, CA 95046
(408) 852 - 0067

System Level Solutions, (India) Pvt, Ltd.

Plot # 32, Zone - D/4, Phase 1,
G.I.D.C. Estate, V.U. Nagar - 388 121
Gujarat, India
91-2692-229280
<http://www.slscorp.com>

Copyright © 2008, System Level Solutions, Inc. (SLS) All rights reserved. SLS, an embedded systems company, the stylized SLS logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of SLS in India and other countries. All other products or service names are the property of their respective holders. SLS products are protected under numerous U.S. and foreign patents and pending applications, mask working rights, and copyrights. SLS reserves the right to make changes to any products and services at any time without notice. SLS assumes no responsibility or liability arising out of the application or use of any information, products, or service described herein except as expressly agreed to in writing by SLS. SLS customers are advised to obtain the latest version of specifications before relying on any published information and before orders for products or services.

ug_neekbsp_1.5

About This Document

This document describes the usage of the uClinux NEEK board support package. With the help of the bsp you can develop embedded applications using NEEK kit and uClinux.

Table below shows the revision history of the document

| Version | Date | Description |
|---------|----------------|--|
| 1.0 | July 2008 | First release |
| 1.1 | July 2008 | Added uClinux demo images and SLS Player images. |
| 1.2 | August 2008 | Remove Chapter 7.0 – Root File System Added section 8.2 quick reference Added section 5.4 I2S Application Added section 3.1.1, 3.1.2, 3.2.1 and 3.2.2 |
| 1.3 | August 2008 | Added Frame buffer console section under 8.2 Quick Reference Added the section 4.5 SLS PS/2 IP Driver |
| 1.4 | August 2008 | Kernel debugging and JFFS2 file system updated |
| 1.5 | September 2008 | Modified document as per kernel version 2.6.27 |

How to Contact SLS

For the most up-to-date information about SLS products, go to the SLS worldwide website at <http://www.slscorp.com>. For additional information about SLS products, consult the source shown below.

| Information Type | Email |
|---|--|
| E-mail Product literature services, SLS literature services, Non-technical customer services, Technical | support@slscorp.com |

Typographic Conventions

This document uses the typographic conventions shown as below.





| Visual Cue | Meaning |
|---|--|
| Bold type with initial capital letters | All headings, subheadings titles in a document are displayed in bold type with initial capital letters. E.g. Configuring and Compiling. |
| Bold | Project names, Menu commands, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: HelloWorld , project name, Exit menu command, romfs/bin directory, sls_neek_bsp_hw_sopc.ptf file. |
| Courier | Anything that must be typed exactly as it appears is shown in Courier type. For example: <code>cd/home/uClinux/uClinux-dist.</code> |
| 1, 2 | Numbered steps are used in a list of items, when the sequence of items is important. Such as steps listed in procedure. |
| • | Bullets are used in a list of items when the sequence of items is not important. |
|  | The hand points to special information that requires special attention |
|  | The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process. |
|  | The warning indicates information that should be read prior to starting or continuing the procedure or processes. |
|  | The feet direct you to more information on a particular topic. |

Table of Contents

| | |
|--|---------------|
| 1. Introduction..... | 1 |
| 1.1 Software..... | 1 |
| 1.2 Hardware..... | 1 |
| 1.2.1 Target processor..... | 1 |
| 1.2.2 NEEK Kit (Nios Embedded Evaluation kit)..... | 1 |
| 1.3 Supported Devices | 2 |
| 1.4 Development Platform Requirements | 2 |
| 1.5 Setup | 2 |
| 2. uClinux NEEK BSP Development Environment..... | 5 |
| 2.1 NEEK BSP SOPC System | 5 |
| 2.2 Development Environment | 5 |
| 2.2.1 NEEK BSP Components | 6 |
| 2.2.2 IP Address Used | 7 |
| 2.3 Development Host | 7 |
| 2.3.1 Downloading and Unpacking uClinux NEEK BSP | 7 |
| 2.3.2 Directory Contents | 8 |
| 2.4 Development Target..... | 8 |
| 2.4.1 Configuring and Compiling the uClinux | 8 |
| 2.4.2 Set System.ptf | 11 |
| 2.4.3 Customization of Kernel Settings | 13 |
| 2.4.4 Building uClinux zImage..... | 14 |
| 3. Downloading and Running zImage | 15 |
| 3.1 Running zImage on Linux..... | 15 |
| 3.1.1 Using JTAG UART Console | 15 |
| 3.1.2 Using Serial UART Console..... | 15 |
| 3.2 Running zImage on windows..... | 17 |
| 3.2.1 Using JTAG UART Console | 17 |
| 3.2.2 Using Serial UART Console..... | 18 |
| 4. Configuring Device Drivers and File Systems..... | 19 |
| 4.1 Board Selection..... | 19 |
| 4.2 Flash Memory (MTD) Driver | 20 |
| 4.2.1 JFFS2 File System Configuration..... | 20 |
| 4.3 SLS SD Card IP Driver..... | 21 |
| 4.3.1 VFAT | 21 |
| 4.4 SLS Ethernet IP Driver | 22 |
| 4.4.1 NFS 23 | 23 |
| 4.5 SLS VGA IP Driver | 23 |
| 4.6 SLS PS/2 IP Driver..... | 23 |
| 4.7 SLS Touch Panel Controller (TPC) IP Driver | 24 |
| 4.8 SLS I ² S IP Driver..... | 24 |
| 4.9 Serial UART Driver..... | 24 |
| 4.10 JTAG UART Driver | 25 |
| 5. User Applications..... | 26 |
| 5.1 Flash and JFFS2 Application | 26 |
| 5.1.1 Flash Tools..... | 26 |

| | |
|--|-----------|
| 5.1.2 Accessing Flash and JFFS2 Applications | 26 |
| 5.1.3 Configuring Flash Partition | 27 |
| 5.2 VGA, LCD and SD Card Application..... | 27 |
| 5.2.1 Viewing the SD Card Images on the VGA and LCD | 28 |
| 5.3 Touch Panel Controller (TPC) Application | 29 |
| 5.4 I ² S Application (To Play MP3 Songs) | 29 |
| 5.5 Nano-x Application..... | 30 |
| 5.6 Adding New User Application..... | 32 |
| 5.7 Build New User Application Using SLS IP Drivers | 33 |
| 5.8 Shell Commands | 33 |
| 6. Configuring Network utilities and NFS (Client) | 35 |
| 6.1 Configuring DHCP Client..... | 35 |
| 6.2 Static IP Allocation | 36 |
| 6.3 Mounting NFS on NEEK..... | 36 |
| 6.3.1 NFS Server (Host) Set Up | 36 |
| 6.3.2 NFS Client (NEEK) Setup | 37 |
| 6.4 Configuring inetd, telnetd, ftpd Server | 37 |
| 6.5 Configuring boa Server | 38 |
| 7. Debugging Kernel and User Application..... | 39 |
| 7.1 Debugging uClinux kernel using nios2-elf-insight over JTAG | 39 |
| 7.2 Debugging User Application using gdbserver over Ethernet | 41 |
| 7.3 Debugging User application using Eclipse IDE..... | 42 |
| 8. Demonstrations & Quick Reference..... | 45 |
| 8.1 Demonstrations | 45 |
| 8.1.1 NEEK Application Selector Demos..... | 45 |
| 8.2 Quick Reference | 52 |
| 8.2.1 Prebuilt_zImage..... | 52 |
| 8.2.2 Framebuffer_console | 52 |

1. Introduction

uClinux NEEK Board support package (BSP) provides developers with the easiest and fastest way to create embedded applications on the NEEK kit (target) using the uClinux operating system. This combination of hardware, firmware, and software form a complete package for building, downloading, and testing applications developed on a host machine:

- Install the uClinux board support package (BSP) on the host machine.
- Connect Ethernet and JTAG cable between host and target for serial communication
- Build the custom Linux kernel for the NEEK board
- Start writing application code.

When ready to see your application in action, just download it to the NEEK board. Now you can test and debug the kernel and application.

1.1 Software

uClinux NEEK BSP is the combination of uClinux, IP core drivers and applications developed by SLS Corporation. Current version of the BSP supports VGA, TPC, SD Card, Ethernet, PS/2 and I²S IP.

uClinux is a port of the Linux for MMU less processor. So many different architectures are supported by uClinux. It has almost all necessary powers as an embedded operating system. This document is based on kernel version 2.6.27. Visit www.uclinux.org for more information.

1.2 Hardware

1.2.1 Target processor

This bsp is targeted for Nios2. Nios2 is a soft core FPGA processor. It is 32 bit RISC general purpose processor. Nios II processor system is equivalent to a microcontroller or “Computer on a chip” that includes a processor and a combination of peripherals and memory on a single chip. The term “Nios II processor system” refers to a Nios II processor core, a set of on-chip peripherals, on-chip memory, and interfaces to off-chip memory, all implemented on a Single Altera device. Like a microcontroller family, all Nios II processor systems use a consistent instruction set and programming model.

1.2.2 NEEK Kit (Nios Embedded Evaluation kit)

The NEEK kit includes feature rich development board having Cyclone III EP3C25F324 FPGA, 256-Mbit DDR SDRAM, 1-Mbyte Synchronous SRAM,

16-Mbytes Intel P30/P33 flash and LCD multimedia card. To get the details of NEEK kit, please read NEEK user guide located at */DOCS*.

1.3 Supported Devices

The NEEK board devices that are currently supported in the uClinux kernel with the uClinux NEEK BSP include:


- Flash
- SD Card IP
- 10/100 Ethernet IP
- VGA/LCD Controller IP
- PS/2 IP
- Touch Panel IP
- I²S IP
- Audio Codec Configuration IP
- Serial UART

1.4 Development Platform Requirements

We assume that user is familiar with Altera tools. Following are Development platform requirements:

- Any Linux distribution having good hardware configuration with at least 2.5 GB free space.
- Altera Development Tools (7.2 or above).

This uClinux BSP is tested on RED Hat Fedora 7 core but it should work on other Linux Desktop distribution also. You can build the kernel and application on Linux desktop only. The images built on Linux can be downloaded and run on Windows if the Altera Development tools are installed.

 This BSP will not work on Cygwin environment.

1.5 Setup

To startup and begin communicating with the board:

1. Establish communication between target (Board) and host (PC) using JTAG cable.
2. Establish a network connection using RJ45 Ethernet cable to debug the application.
3. Establish a serial connection using RS232 cable to use Serial UART as console.

4. Apply power to the NEEK board.
5. Connect VGA connector of the monitor to the VGA port of NEEK board if you want to see the images on both the VGA screen and LCD.

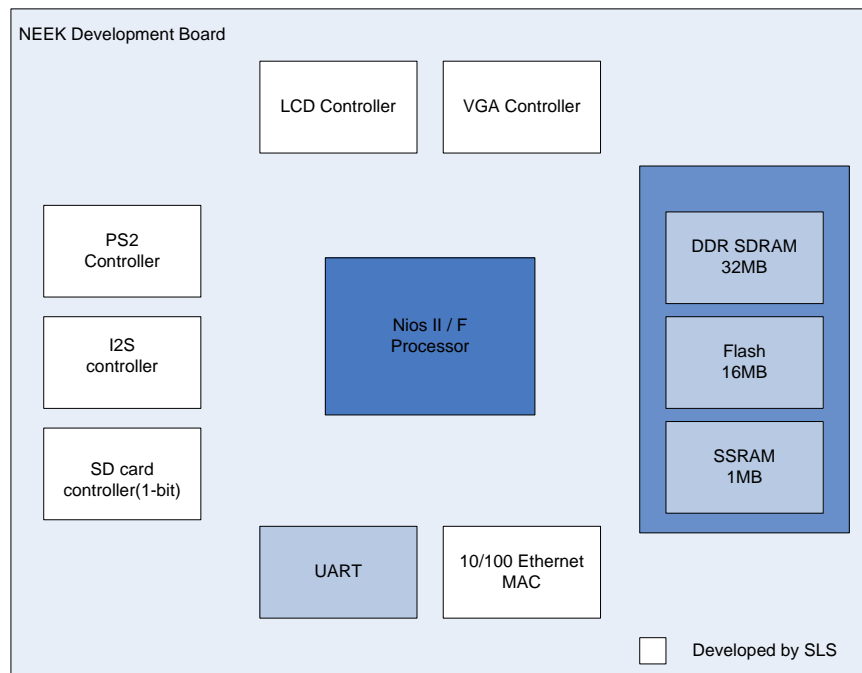
2. uClinux NEEK BSP Development Environment

This chapter provides information to help setup the development environment for the NEEK board.

2.1 NEEK BSP SOPC System

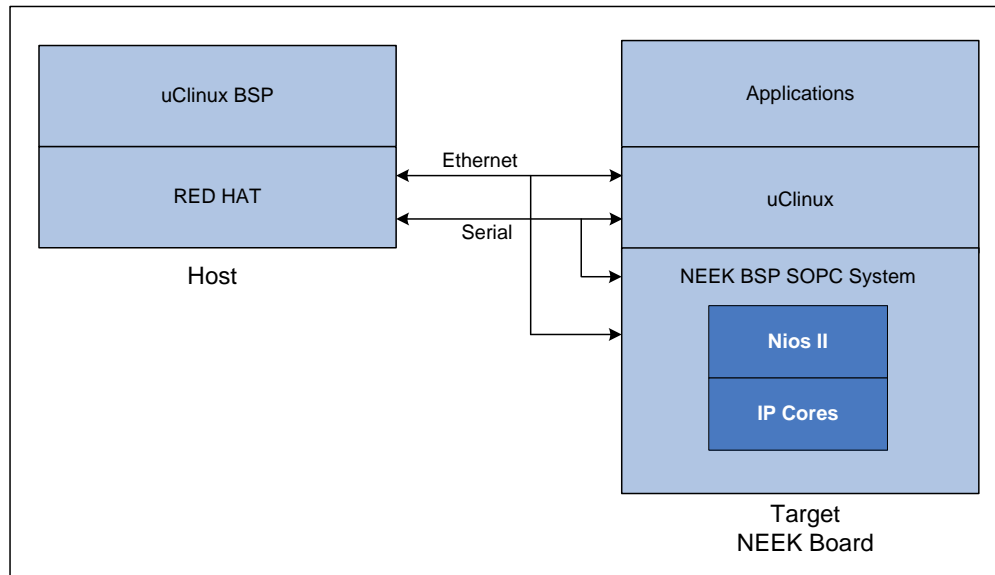
The [Figure 1](#) below shows the block diagram of NEEK BSP SOPC system.

Figure 1 : NEEK BSP SOPC System



2.2 Development Environment

The Nios II embedded development environment consists of two systems: a host system and a target system. The host system is used for compiling, linking, remote debugging, and associated development activities. The target system, such as the NEEK board, is used for application development and testing ([Figure 2](#)). The board acts as a target for application development

Figure 2 : Development Environment

2.2.1 NEEK BSP Components

Table below lists the components included in the uClinux NEEK BSP.

| Component | Description |
|------------------------|-------------|
| kernel | v2.6.27 |
| gcc | v3.4.6 |
| Ethernet driver | Included |
| serial driver | Included |
| ps2 keyboard | Included |
| Touch panel controller | Included |
| VGA/LCD driver | Included |
| SDcard driver | Included |
| I2s driver | Included |
| Audio codec driver | Included |
| FLASH driver | Included |
| JFFS2 support | Included |
| VFAT support | Included |
| NFS support | Included |
| MP3 support | Included |
| JPEG support | Included |

2.2.2 IP Address Used

The table below lists the IP address and port used throughout this document. It may be different for your system. Please check your network settings before applying any IP address.

| IP Address | Description |
|---------------|--|
| 192.168.0.14 | Development Target IP address |
| 192.168.0.26 | Development Host and NFS server IP address |
| 192.168.0.205 | Getway IP Address |
| 255.255.255.0 | Getway Mask |
| 9999 | TCP port |

2.3 Development Host

Refer to the Red Hat documentation for Red Hat Fedora 7.0 core installation and host setup information.

2.3.1 Downloading and Unpacking uClinux NEEK BSP

To download the uClinux NEEK BSP package visit:
<http://slscorp.com/pages/bspdownload.php>.

Now, to unpack the BSP package, follow the steps mentioned below:

1. Copy the entire package in the home directory.
2. Apply **su** command to login as super user

```
#su
```

It will ask super user password for further process.

3. Issue

```
#cd /home
```

```
#tar jxvf uClinux.tar.bz2
```

4. Now go into uClinux directory using the following command

```
#cd /home/uClinux
```

And then

```
#tar jxvf uClinux-dist.tar.bz2
```

It starts extracting files and it may take some time.

5. Now, issue

```
#cd Bintools
#tar jxvf nios2gcc-20080203.tar.bz2
```

6. Return to uClinux

```
#cd..
```

7. Issue command

```
#ls
```

Here you will see nios2-linux Bintools System-files.

2.3.2 Directory Contents

You have now successfully unpacked the BSP package. You will see following directory structure:

| Directory | Description |
|-----------------|---|
| uClinux | Contains uClinux-dist, Bintool (binary tool chain) and system-files. User who is interested only in software side can start working with this directory. |
| docs | Contains BSP documents: uClinux NEEK BSP user guide, NEEK user guide, DRIVERSAPI.txt and boot_msg.txt. |
| demos | Contains SLS uClinux Demo and Format utility for SD card. In each subdirectory you will find other necessary information like quick start guide and readme. |
| Ref_design | Contains Quartus Archive project of SOPC system used in this BSP |
| Quick Reference | Pre-built zImage for ready reference. |

2.4 Development Target

2.4.1 Configuring and Compiling the uClinux

NEEK BSP has two main areas of configurations. Together they define a complete uClinux platform configuration. The two areas include:

- Vendor / Product Settings
- Kernel Settings

In this section, you will learn all the above listed configurations.

Start the Configuration Menu

To configure the kernel, go to the directory “/home/uClinux/nios2-linux/uClinux-dist” and follow the steps below:

1. Open the Linux terminal.
2. On the terminal, change into the \$home/uClinux/uClinux-dist directory and change the environment path. Use one of the four main kernel configuration methods to start the configuration menu.

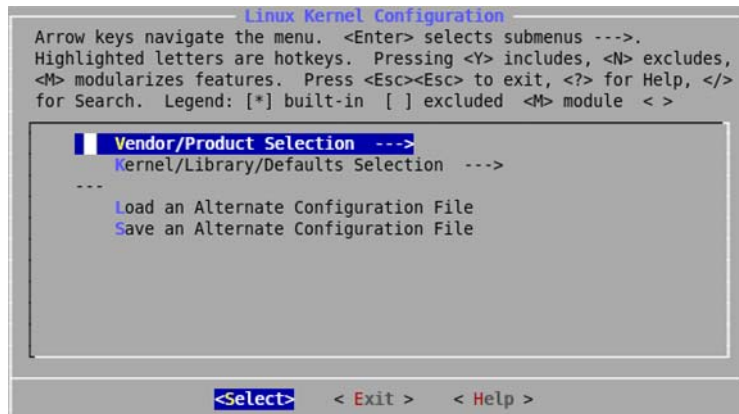
```
#cd /home/uClinux/nios2-linux/uClinux-dist
#PATH=$PATH:/home/uClinux/Bintools/opt/nios2/bin
```

6. Configure the kernel

```
#make clean
#make menuconfig
```

You will see the Kernel configuration main menu as shown in [Figure 3](#).

Figure 3 : Linux Kernel Configuration Main Menu

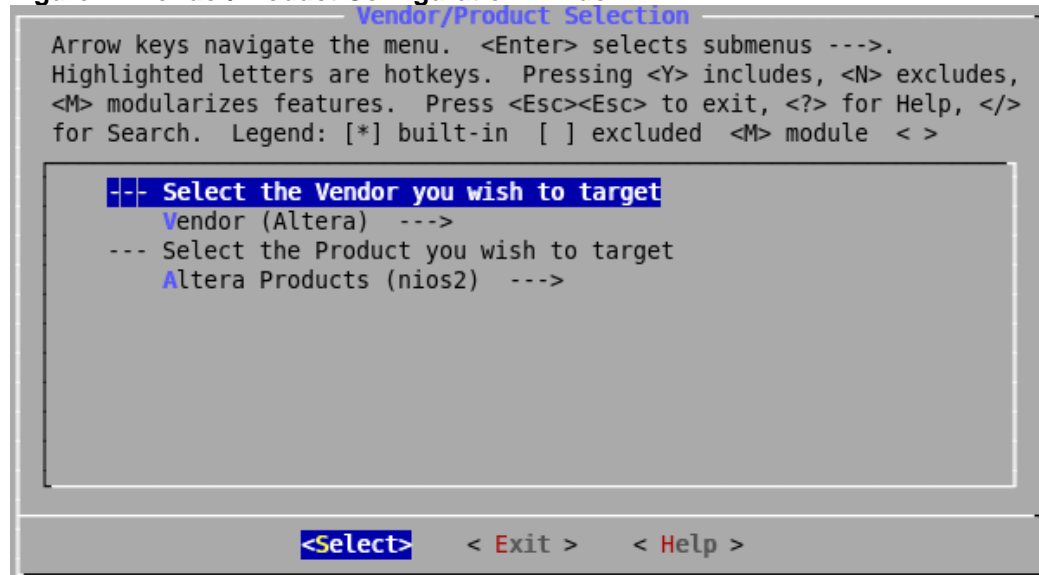


Use one of the two main kernel configuration methods to start the configuration.

- Use \uparrow or \downarrow arrow to select the menu item and then press \leftarrow to go into the submenu. Use \leftarrow or \rightarrow and then press \leftarrow to select the menu command “<Select>” “<Exit>” or “<Help>” throughout the configuration.

Configuring Vendor/Product Settings

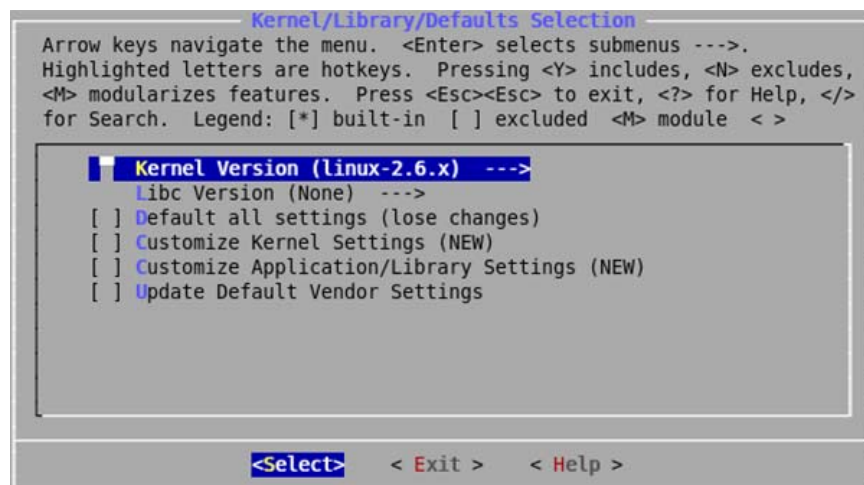
1. Select the submenu **Vendor/Product Selection --->** from the [Figure 3](#). press **Enter**. You will see Vendor/Product Configuration window. See [Figure 4](#)

Figure 4 : Vendor/Product Configuration Window

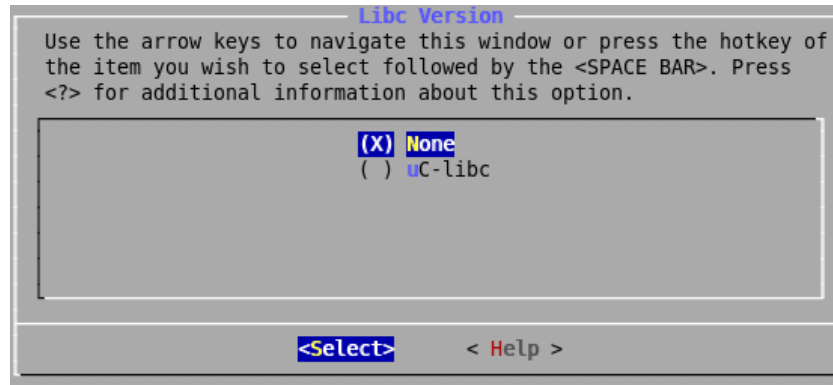
2. Select the following options:
 - Vendor: **Vendor (Altera)**
 - Target Product: **Altera Products (nios2) ---->**
3. Select **<Exit>**. You will return to Kernel configuration menu [Figure 3](#).

Configuring Kernel/Library/Defaults Settings

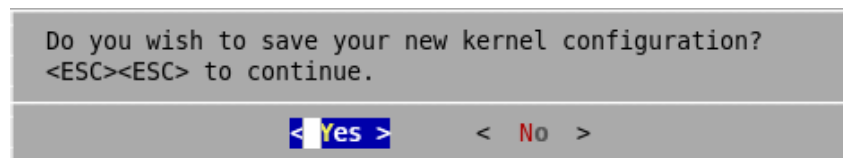
1. Select the option **Kernel/Library/Defaults Selection --->** from the Kernel Configuration Menu [Figure 3](#). You will see the Kernel/Library/Defaults Selection submenu as shown in [Figure 5](#)

Figure 5 : Kernel/Library/Defaults Selection Window

2. Select **Libc Version (None) ---->**. You will see the as shown in [Figure 6](#).


Figure 6 Libc Version Settings

3. Press **Enter** to select **None**.
4. Select **<Exit>**
5. Select **<Exit>**
6. You will be asked to save the kernel configuration. See [Figure 7](#)

Figure 7 : Save Option

7. Select **<Yes>** and press **Enter**.

You have now finished Kernel/Library/Defaults settings.

-  DO NOT change any other settings until first successful boot.
8. You will be backed to Linux Terminal.

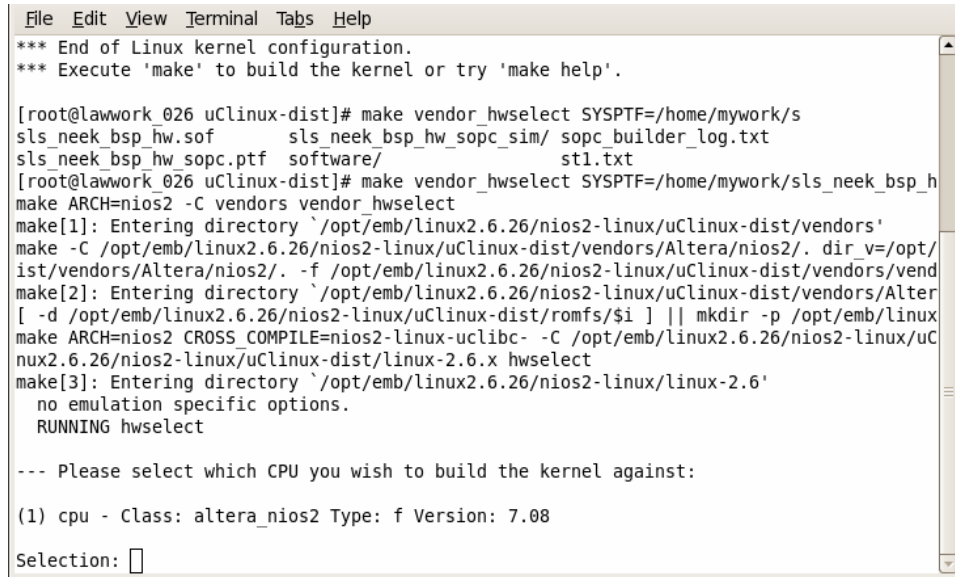
2.4.2 Set System.ptf

Generate a system header file with the **sls_neek_bsp_hw_sopc.ptf**. This file is located at **“/home/uClinux/System-files”** directory. You will be prompted to select the hardware from the available list.

1. Type the following command and press **Enter**.

```
#make vendor_hwselect SYSPTF=/home/uClinux/
System-files/sls_neek_bsp_hw_sopc.ptf
```

You will be asked to select the CPU. See [Figure 8](#)

Figure 8 : CPU Selection


```

File Edit View Terminal Tabs Help
*** End of Linux kernel configuration.
*** Execute 'make' to build the kernel or try 'make help'.

[root@lawwork_026 uClinux-dist]# make vendor_hwselect SYSPTF=/home/mywork/s
sls_neek_bsp_hw.sof      sls_neek_bsp_hw_sopc_sim/ socp_builder_log.txt
sls_neek_bsp_hw_sopc.ptf software/          st1.txt
[root@lawwork_026 uClinux-dist]# make vendor_hwselect SYSPTF=/home/mywork/sls_neek_bsp_h
make ARCH=nios2 -C vendors vendor_hwselect
make[1]: Entering directory `/opt/emb/linux2.6.26/nios2-linux/uClinux-dist/vendors'
make -C /opt/emb/linux2.6.26/nios2-linux/uClinux-dist/vendors/Altera/nios2/. dir_v=/opt/
ist/vendors/Altera/nios2/. -f /opt/emb/linux2.6.26/nios2-linux/uClinux-dist/vendors/vend
make[2]: Entering directory `/opt/emb/linux2.6.26/nios2-linux/uClinux-dist/vendors/Alter
[ -d /opt/emb/linux2.6.26/nios2-linux/uClinux-dist/romfs/$i ] || mkdir -p /opt/emb/linux
make ARCH=nios2 CROSS_COMPILE=nios2-linux-uclibc- -C /opt/emb/linux2.6.26/nios2-linux/uC
nux2.6.26/nios2-linux/uClinux-dist/linux-2.6.x hwselect
make[3]: Entering directory `/opt/emb/linux2.6.26/nios2-linux/linux-2.6'
no emulation specific options.
RUNNING hwselect

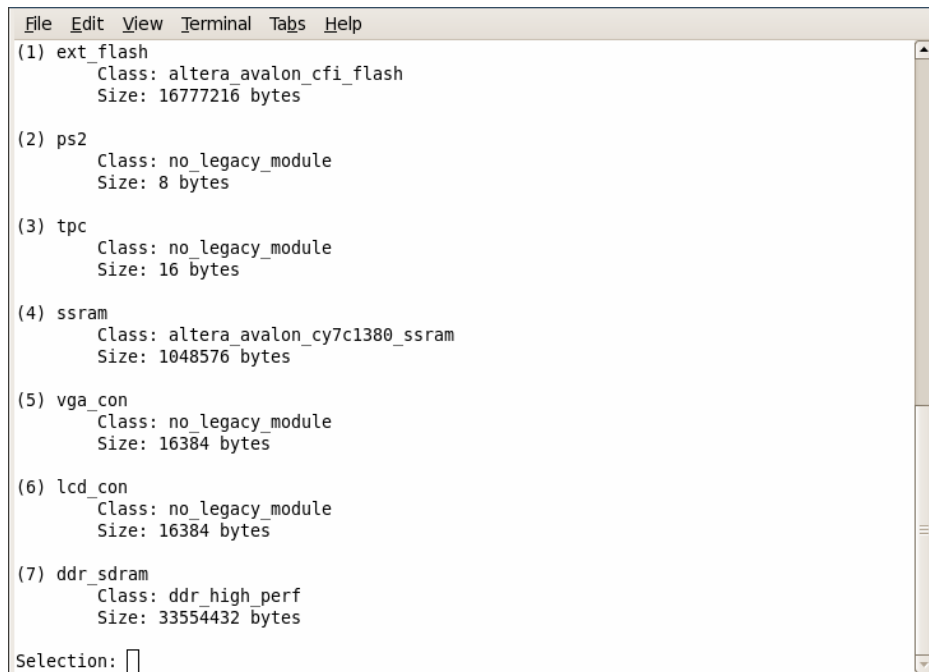
--- Please select which CPU you wish to build the kernel against:

(1) cpu - Class: altera_nios2 Type: f Version: 7.08

Selection: 

```

2. Enter the choice **(1)**. It will ask to select a device to execute kernel from:
See [Figure 9](#)

Figure 9 : Select a Device to Execute Kernel From


```

File Edit View Terminal Tabs Help
(1) ext_flash
    Class: altera_avalon_cfi_flash
    Size: 16777216 bytes

(2) ps2
    Class: no_legacy_module
    Size: 8 bytes

(3) tpc
    Class: no_legacy_module
    Size: 16 bytes

(4) ssram
    Class: altera_avalon_cy7c1380_ssram
    Size: 1048576 bytes

(5) vga_con
    Class: no_legacy_module
    Size: 16384 bytes

(6) lcd_con
    Class: no_legacy_module
    Size: 16384 bytes

(7) ddr_sdram
    Class: ddr_high_perf
    Size: 33554432 bytes

Selection: 

```

7. To select DDR SDRAM, enter:
Selection:7
8. Now create romfs directory by issuing following command.

```
#make romfs
```

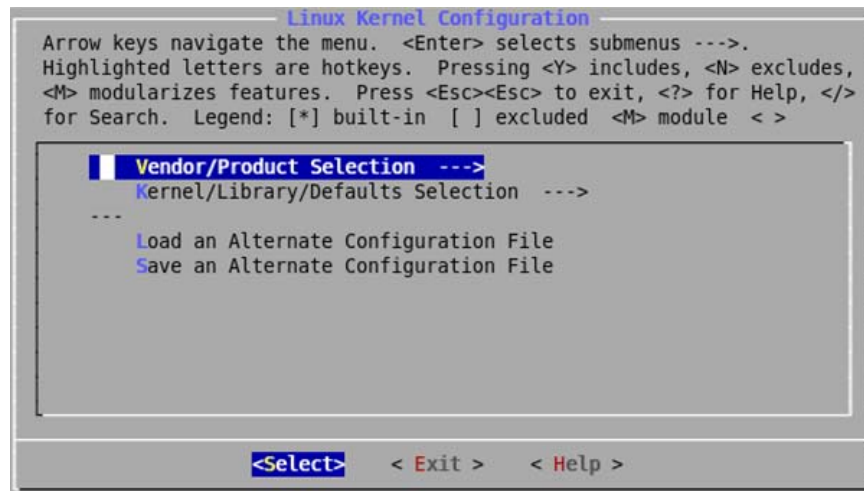
You may get error. Ignore it.

2.4.3 Customization of Kernel Settings

1. To customize the Kernel Settings, type on the terminal:

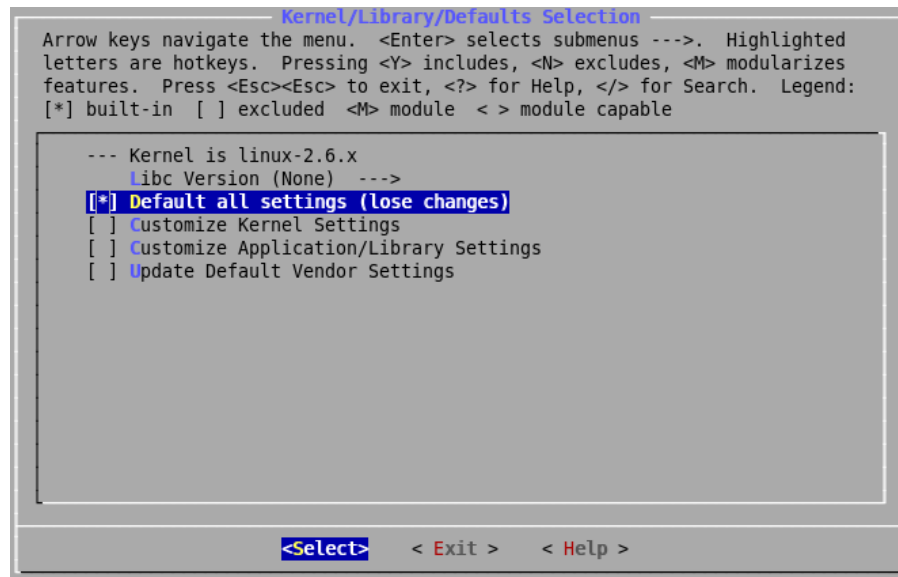
```
#make menuconfig
```

Figure 10 : Linux Kernel Configuration Window



2. Select **Kernel/Library/Defaults Selection** and press **Enter**. You will see the more kernel configuration settings as shown in [Figure 11](#).

Figure 11 : Customize Kernel/Library/Defaults Selection Window



3. Select Default all settings by pressing “Y” :
4. Select **<exit>**.

5. Select **<exit>**

6. Select **<yes>** for saving all changed settings.

Save all the settings and exit.

2.4.4 Building uClinux zImage

Once you configured the kernel, build uClinux Image by issuing following commands.

```
#make
```

Here you will get **zImage** (elf file) at
/home/ uClinux/ nios2-linux/uClinux-dist/image

3. Downloading and Running zImage

zImage is one type of elf file which contains compress kernel image and romfs image. Given below are the steps for downloading a running zImage in to the NEEK hardware platform.

3.1 Running zImage on Linux

3.1.1 Using JTAG UART Console

To run zImage on Linux, you should have Nios2 EDS installed on your pc with properly set environmental variables. We will use utilities from Nios2 EDS. This description is for JTAG UART console only.

First configure .sof

```
#nios2-configure-sof /home/uClinux/System-  
files/sls_neek_bsp_hw.sof
```

To download zImage, open the terminal

```
#nios2-download -g /home/uClinux/nios2-linux/uClinux-dist  
/images/zImage
```

Run the command

```
#nios2-terminal
```

Here you will see booting.

3.1.2 Using Serial UART Console

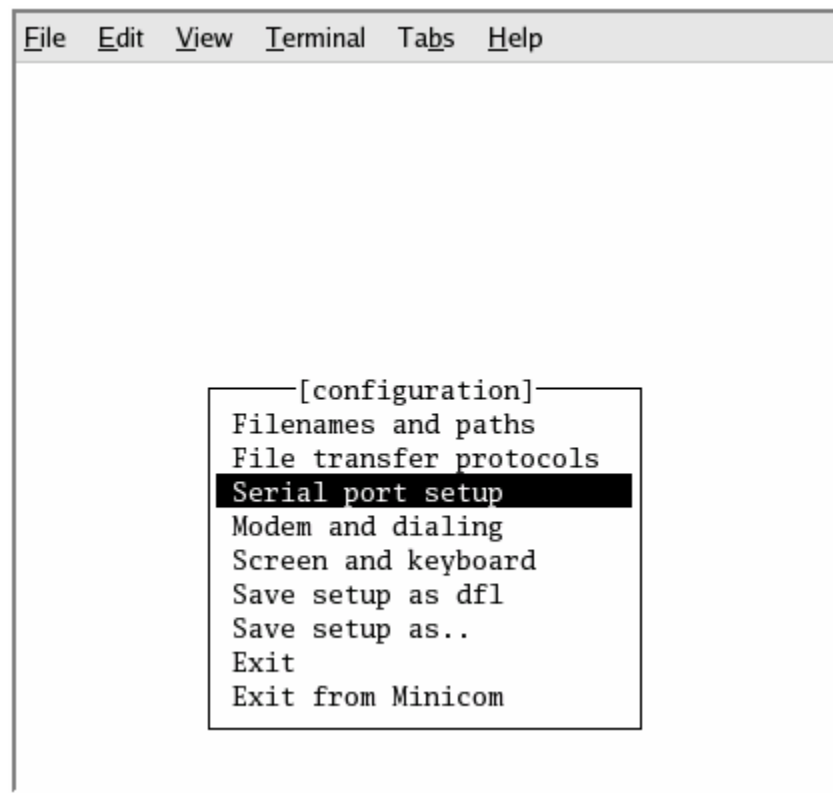
Use Minicom as serial terminal emulator. First of all configure Minicom to use it with zImage. Follow the steps below to configure Minicom.

1. On your terminal type,

```
# minicom -s
```

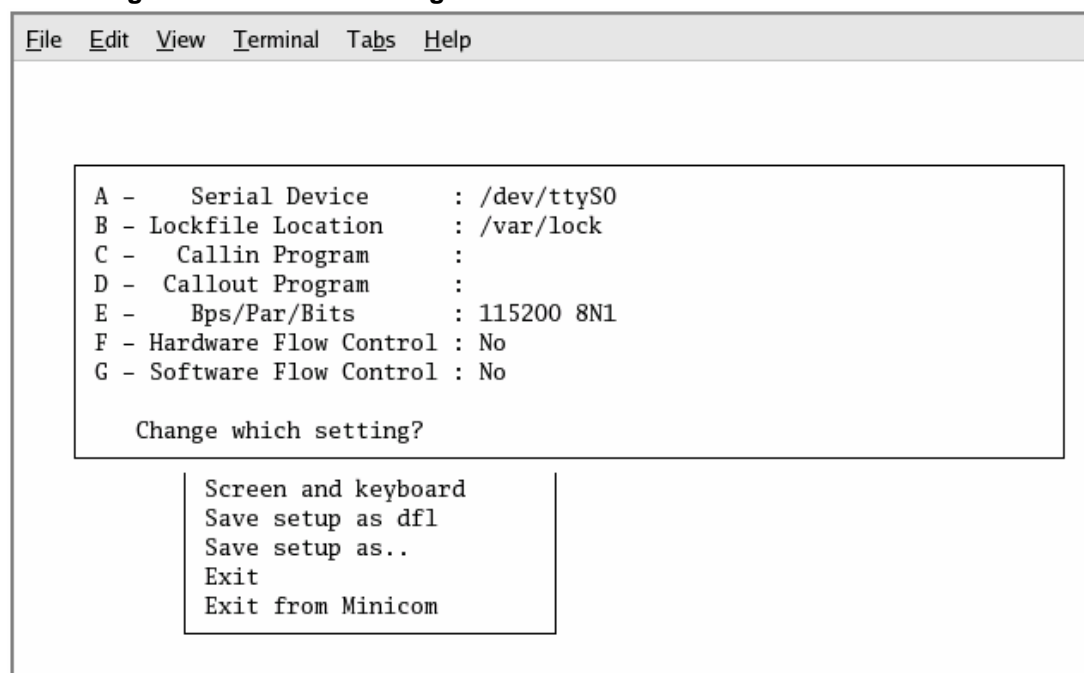
The Minicom configuration window opens. See [Figure 12](#) . Select **Serial port setup** and press **Enter** key.

Figure 12 Minicom Configuration Window1



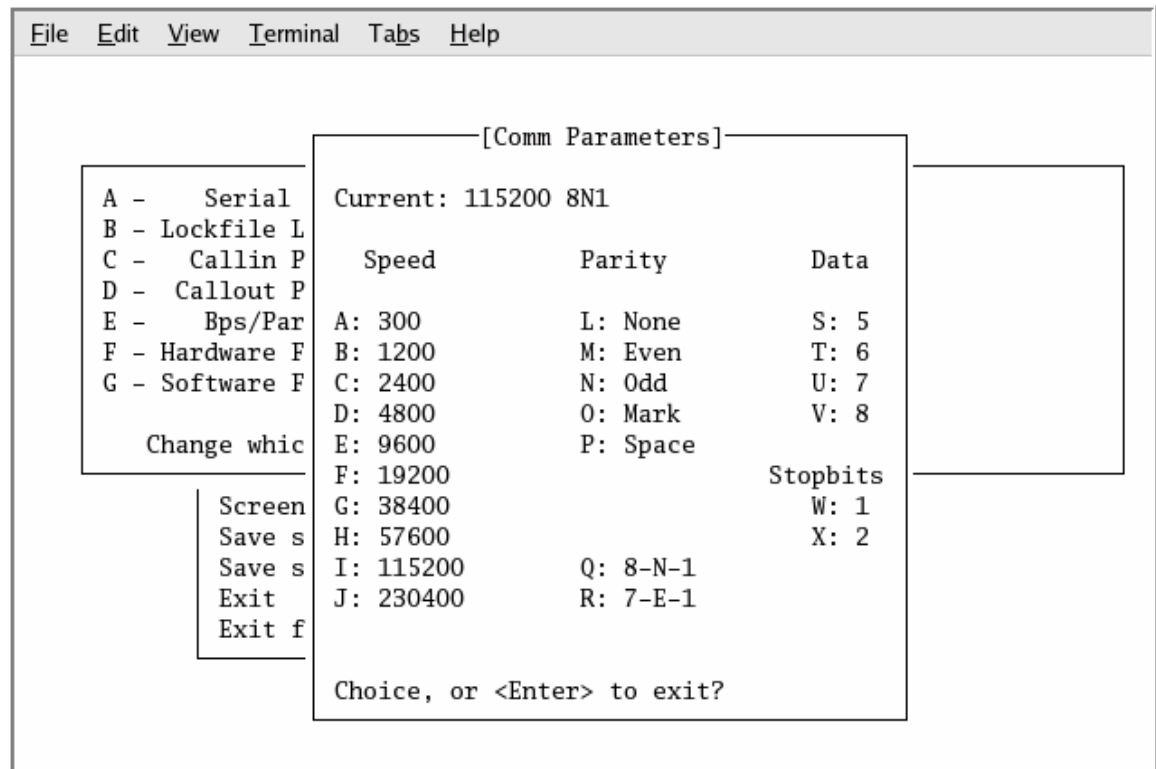
2. Select the Serial Device, baud rate and other parameters as shown in [Figure 13](#). Here, we have selected Serial device as **/dev/ttyS0** but it may differ as per your development machine.

Figure 13 : Minicom Configuration Window 2



3. Set the baud rate 115200, 8 bit, no parity and one stop bit. See the [Figure 14](#)

Figure 14 : Minicom Configuration Window3



4. You may set this configuration as default by selecting **Save Setup as dfl** option after all configurations.
5. Now, download .sof and zImage from other terminal and you will see booting on Minicom.

3.2 Running zImage on windows

3.2.1 Using JTAG UART Console

If you have selected **JTAG UART** as console while building the zImage then follow the steps:

```
[SOPC Builder]$ nios2-configure-sof
s1s_neek_bsp_hw.sof

[SOPC Builder]$ nios2-download -c USB-Blaster[USB-0]
zImage -g

[SOPC Builder]$ nios2-terminal
```

Here you can see Linux booting messages on same shell.

3.2.2 Using Serial UART Console

You can build zImage on Linux and run it on windows. This is the description for serial UART console. Go to the "/home/uClinux/uClinux-dist/images" directory. Copy the file named "zImage" and paste it in to your windows PC (make sure your zImage absolute path doesn't contain a white character i.e. space).also copy **sls_neek_bsp_hw.sof** from /home/uClinux/System-files into windows directory.

1. Open the HyperTerminal with the following configuration.

Figure 15 : COM1 Properties




2. Download **sls_neek_bsp_hw.sof** file into the NEEK either Quartus programmer or with command.
3. Open the Nios II Command Shell. Navigate to the directory where you have put the zImage and sof files.

```
[SOPC Builder]$ cd /(your directory containing zImage)
[SOPC Builder]$ nios2-configure-sof
sls_neek_bsp_hw.sof
[SOPC Builder]$ nios2-download -c USB-Blaster[USB-0]
zImage -g
```

Here you can see Linux booting messages on the hyper terminal window.

4. Configuring Device Drivers and File Systems

If more functions need to be supported on kernel, then before the kernel rebuild, it needs to do the selected kernel configuration, the following sessions describe the procedures to do configurations.

-  To know more about peripherals and hardware available on the NEEK kit, refer the NEEK Kit Getting Started User Guide located at <BSP Installation Path>/Docs.

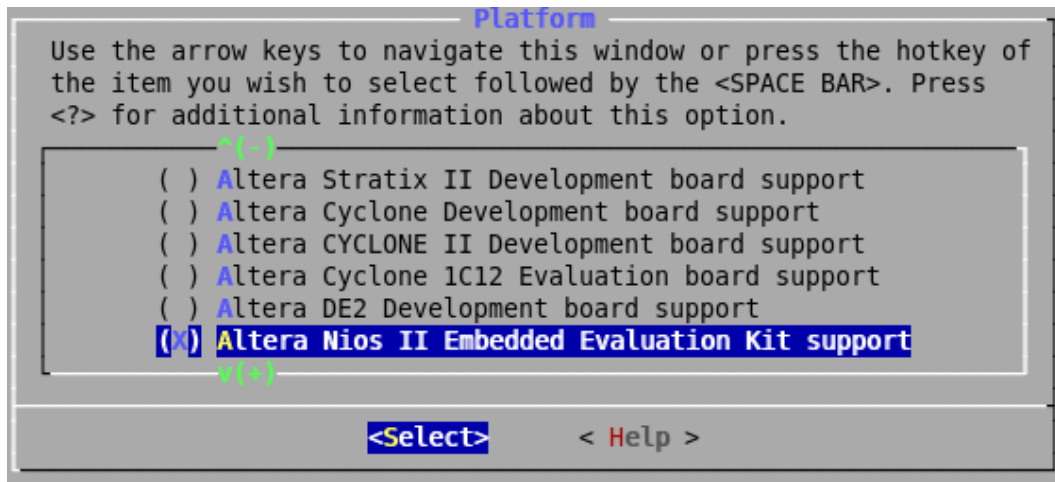
4.1 Board Selection

From the kernel settings, select the following options.

```
Processor type and features --->
    Platform (Default Board) --->
```

You will see the board selection options as shown in [Figure 16](#). Select the **NEEK Kit** option and press **Enter**.

Figure 16 : Board Selection



4.2 Flash Memory (MTD) Driver

To include the Flash Memory (Memory Technology Device) driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Memory Technology Devices (MTD) --->
    [*] Memory Technology Device (MTD) support
    [*] MTD partitioning support
    [*] Direct char device access to MTD devices
    [*] Caching block device access to MTD devices
  RAM/ROM/Flash chip drivers --->
    [*] Detect flash chips by Common Flash
        Interface (CFI) probe
    [*] Support for Intel/Sharp flash chips
  Mapping drivers for chip access --->
    [*] CFI Flash device in physical memory map
        (0x8000000) Physical start address of flash
        mapping (NEW)
    (0) Physical length of flash mapping (NEW)
    (2) Bank width in octets (NEW)
```

4.2.1 JFFS2 File System Configuration

JFFS2 is a log-structured file system designed for use on flash devices in embedded systems. Rather than using a kind of translation layer on flash devices to emulate a normal hard drive, as is the case with older flash solutions, it places the file system directly on the flash chips.

If you want to use JFFS2 file system on Flash Memory then you have to configure both the Flash Memory driver and JFFS2 file system. First do the Flash Memory Driver configuration as described above. And then follow the procedure below to configure the JFFS2 File system.

```
File systems --->
  Miscellaneous File systems --->
    [*] Journaling Flash File System v2 (JFFS2)
    support
    (0) JFFS2 debugging verbosity (0 = quiet, 2 =
    noisy)
    [*] JFFS2 write-buffering support
```

4.3 SLS SD Card IP Driver

To include the SD Card Driver in compilation, the following options should be enabled:

```
Device Drivers --->
  [*] MMC/SD card support --->
    --- MMC/SD card support
    --- MMC/SD Card Drivers
    [ ] MMC block device driver
    --- MMC/SD Host Controller Drivers
    [*] SLS SD card device driver
```

4.3.1 VFAT

Virtual File Allocation Table (VFAT) is the part of the Windows 95 and later operating system that handles long file names, which otherwise could not be handled by the original file allocation table file allocation table (FAT) programming.

If you want to use VFAT file system on SD Card then you have to configure both the SD Card driver and VFAT file system. First do the SD Card Driver configuration as described above. And then follow the procedure below to configure the VFAT File system.

```
File systems --->
  DOS/FAT/NT File systems --->
    [*] MSDOS fs support
    [*] VFAT (Windows-95) fs support
    (850) Default codepage for FAT
    (Iso8859-1) Default iocharset for FAT
    Native Language Support

    --- Base native language support
      (iso8859-1) Default NLS Option
      [*] Codepage 437 (United States, Canada)
      [*] Codepage 850 (Europe)
      [*] NLS ISO 8859-1 (Latin 1; Western European
          Languages)
      [*] NLS UTF-8
```

4.4 SLS Ethernet IP Driver

To include the SLS Ethernet IP Driver in compilation, the following options should be enabled:

```
Networking --->
  [*] Networking support
  --- Network device support
    [ ] Network packet debugging (NEW)
    [*] Packet socket
    [ ] Packet socket: mmaped IO (NEW)
    [*] UNIX domain sockets
    [ ] PF_KEY sockets (NEW)
    [*] TCP/IP networking

Device Drivers --->
  [*] Network device support --->
    Ethernet (10 or 100Mbit) --->
      [*] SLS MAC support
```

4.4.1 NFS

NFS is a network file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network as easily as if the network devices were attached to its local disks.

If you want to use NFS file system on Ethernet then you have to configure both the Ethernet IP driver (as described above) and NFS file system. Then follow the procedure below to configure the NFS File system

```
File systems -->
  Network File Systems -->
    [*] NFS file system support
    [*] Provide NFSv3 client support
```

4.5 SLS VGA IP Driver

To include the SLS VGA IP Driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Graphics support --->
    [ ] Enable firmware EDID
    [*] Support for frame buffer devices --->
      --- Frame buffer hardware drivers
        [*] SLS VGA IP support
```

4.6 SLS PS/2 IP Driver

To include the SLS PS/2 IP Driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Input device support --->
    [*] Generic input layer (needed for keyboard,
    mouse...)
    --- Input Device Drivers
    [*] Keyboards --->
      --- Keyboards
        [*] SLS PS2 Keyboard --->
```

4.7 SLS Touch Panel Controller (TPC) IP Driver

To include the SLS TPC IP Driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Character devices --->
    [*] SLS TPC support
```

4.8 SLS I²S IP Driver

To include the SLS I²S IP driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Character devices --->
    [*] SLS I2S support
```

4.9 Serial UART Driver

To include the Serial UART Driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Character devices --->
    Serial support --->
      [*] Altera UART support
```

If you want to select console on Serial UART, select following option:

```
Device Drivers --->
  Character devices --->
    Serial support --->
      [*] Altera UART support
      [*] Altera UART console support
```

If you need more than one UART support, select following option:


```
Device Drivers --->
  Character devices --->
    Serial support --->
      [*] Altera UART support
          (4) Maximum number of Altera uart ports
          (115200) Default baudrate for Altera UART
                ports
```

4.10 JTAG UART Driver

To include the JTAG UART Driver in compilation, the following options should be enabled:

```
Device Drivers --->
  Character devices --->
    Serial support --->
      [*] Altera JTAG UART support
      [*] Altera JTAG UART console support
```

If you want to bypass the JTAG, then select the following options.

```
Device Drivers --->
  Character devices --->
    Serial support --->
      [*] Altera JTAG UART support
      [*] Altera JTAG UART console support
      [*] Bypass output when no connection
```

5. User Applications

Using the User Applications you can access the drivers, file systems and peripherals available on the board. This chapter describes the following user applications.

5.1 Flash and JFFS2 Application

To access Flash and JFFS2 application, include its driver and file system as described in the section [Flash Memory \(MTD\) Driver](#) and [JFFS2 File System Configuration](#).

5.1.1 Flash Tools

Select **Customize Application/Library Settings>Flash tools**. Select the options as shown below.

```
Flash Tools --->
  --- MTD utils
  [*] mtd-utils
  [*] erase
  [*] eraseall
  [*] unlock
  [*] lock
  [*] mkfs.jff2
```

Build the kernel as explained in the section [Development Target](#).

5.1.2 Accessing Flash and JFFS2 Applications

To access the Flash and JFFS2 in your application use the required commands as described in the table below:

Table below lists Flash and JFFS2 commands along with the description.

| Command | Description |
|-------------------------------|--|
| <code>cat /proc/mtd</code> | Displays flash partitions. |
| <code>cat /proc/mounts</code> | Displays already mounted files. |
| <code>ls -l /dev/mtd*</code> | Lists mtdblocks and other details. |
| <code>unlock /dev/mtdx</code> | Unlocks all the sectors of mtd device. e.g.: <code>unlock /dev/mtd0</code> , this will unlock all sectors in mtd device0. |

| | |
|--|---|
| flash_erase device offset number_of_blocks | Erases number of blocks of a device starting from the given address. e.g. : flash_erase /dev/mtd0 0x00000 5 .This command would erase 5 blocks of mtdblock0 starting from the offset address 0x00000. |
| flash_eraseall /dev/mtdx | Erases all the contents of mtd device. e.g. : flash_eraseall /dev/mtd0. This would erases all the content from mtd device 0. Note: here mtd0 means mtd device0 and not mtdblock0. If you are not able to erase the content and get the message “ read only file system ” then use the unlock command mentioned above and after that use eraseall. |
| mount -t jffs2 dev/mtdblockx /mnt | Mounts mtdblockx partition on the /mnt directory. You can also use other directory than /mnt e.g. : mount -t jffs2 /dev/mtdblock2 /mnt This would mount mtdblock2 on the mnt directory. Note: Erase entire flash before mounting JFFS2 file system. |
| mkdir new | Creates a new directory |
| cp /new /dev/mtdx | Copies a new directory to mtdx. e.g. : cp /new /dev/mtd0 |
| lock /dev/mtdx | Locks all sectors of the mtd device. e.g. : lock /dev/mtd0 This would locks all sectors of mtd device0. |
| Note: (1). Where x=0, 1, 2.. mtd device or mtdblocks in the system. (2). By default sectors of flash device are locked. So you can not write anything before unlocking the flash device. Use unlock command to write on flash device. | |

5.1.3 Configuring Flash Partition

To configure the flash partition, either you can create your own mapping driver or modify the partition with **config.c** file. Here **config.c** is the mapping driver for mtd device. It is located at *<path_to_dist>/linux-2.6/arch/nios2/kernel*.

5.2 VGA, LCD and SD Card Application

To access the VGA, LCD and SD Card applications, select necessary drivers and file systems for SD Card, VGA and LCD as mentioned in the [SLS SD Card IP Driver](#), [VFAT](#) and [SLS VGA IP Driver](#).sections.


Before going further, copy .jpg images from your PC in the SD Card and make the following selection.

```
Miscellaneous Applications --->
  ---video tools
    [*] jpegview
```

5.2.1 Viewing the SD Card Images on the VGA and LCD

1. To mount the SD Card on mnt directory, Issue the following command on the target when the system is up and running.

```
/>mount -t vfat /dev/hda /mnt ↵
```

 Sometimes, when the user uses this command on the console, it is failed to mount and returns with following error message.

```
FAT: bogus number of reserved sectors
```

```
VFS: Can't find a valid FAT file system on dev SLS_SD
```

This is due to the problem in the file system formatted on the SD Card, which is supported in windows but is not supported in uClinux. To solve this problem, please download [Quick Reference](#).

When you extract Quick Reference, you will find following files:

- zImage
- sls_neek_bsp_hw.sof
- readme.txt

Download .sof and zImage respectively. You will get console on serial UART. When zImage is up and running, plug the SD card into socket of the NEEK board. Issue following command to format the SD card.

```
/>mkdosfs -I -F 16 /dev/hda
```

After formatting the SD card, you have to copy the JPG file from your PC through card reader in the NEEK board. Once again mount the SD card with VFAT file system.

3. Go to the mnt directory and see the contents of the SD card by Issuing the following command.

```
/>cd mnt ↵
```

```
/mnt>ls ↵
```

You will see on the console, the list of images contained in the SD card.

```
10036828.jpg
```

```
10176026.jpg
```

```
dho13v.jpg
```

4. View the images contained in the SD card on VGA monitor connected to VGA port of NEEK board and NEEK board LCD screen by issuing the following command.

```
/mnt>jpegview -S1 -f 10036828.jpg 10176026.jpg ↵
```

The console displays:

```
SLS_VGA Driver is opened
```

```
800 480 800 480 0 0 16 0 -1 -1
```

```
framebase = 0x5000000 err=0
```

```
ImageWidth=732 ImageHeight=480
```

```
read 10036828.jpg OK
```

```
ImageWidth=719 ImageHeight=480
```

```
read 10176026.jpg OK
```

5.3 Touch Panel Controller (TPC) Application

To access the Touch Panel Controller applications, select TPC driver as mentioned in section [SLS Touch Panel Controller \(TPC\) IP Driver](#).

Select the user application as shown below:

```
Miscellaneous Applications --->
[*] userapp
```

Build the kernel with this application. When the kernel is up and running, execute the command:

```
/>touchpanel & ↵
```

Now touch any part of LCD screen and you will see its co-ordinates get printed on the console.

5.4 I²S Application (To Play MP3 Songs)

To play MP3 songs with I²S IP core, please select [SLS I2S IP Driver](#). In miscellaneous application, select following:

```
miscellaneous application -->
  --- Audio tools
  [*] mp3play
```

Build the zImage with these options. To test this application, copy the MP3 songs in SD card and run the zImage, when it is up and running. Mount the SD card as explained in [Viewing the SD Card Images on the VGA and LCD](#). Issue following command:

```
/> cd mnt
mnt> ls
sample.mp3
.....
.....
mnt>mp3play sample.mp3
```

Connect the headphone with the board and you will hear an mp3 song now. You can play any mp3 song with its name as explained above.

5.5 Nano-x Application

To support Nano-X, select Touch Panel Controller driver and VGA driver as mentioned in [SLS Touch Panel Controller \(TPC\) IP Driver](#) & [SLS VGA IP Driver](#).

Select the following options:

```
MicroWindows --->
  [*] MicroWindows
  --- Compiling Options
  [*] Optimize
  [*] Thread safe
  --- Libraries
```

```
[*] NanoX
--- Demos
[*] NanoXDemo
--- Applications
[*] NanoWM
--- Settings
Screen PixType (Packed-16bit-5/6/5) --->
[*] Have File IO
[*] Have BMP Support
[*] Have GIF Support
[*] Have PNM Support
[*] Have JPEG Support
[*] Have PNG Support
--- Display Config
[*] Frame Buffer Display
--- Mouse/Touch Screen
[*] SLS TPC support
--- Keyboard
[*] No Keyboard
--- Install These Applications
[*] Nano-X
[*] NanoWM
[*] NXclock
```

Type following to run the nano-X server,

```
/> nano-X &
/> nanowm &
/> nxclock &
```

It should display a round clock on the LCD screen.

5.6 Adding New User Application

This section explains you of adding a user application named **hello** in uClinux. Follow the steps below to add a new user application.

1. Create **hello** directory in the */home/uClinux/uClinux-dist/user* directory.
2. Copy source file (C file) to the hello directory
3. Add the configuration variable **hello** to the *user/Kconfig* file:

```
config USER_HELLO
bool "hello"
help
help_words_here.....
```

This adds the **hello** menu option to the userland configuration menu.

4. Add following lines to the *user/Makefile* to add the hello directory in compilation.

```
...
dir_${CONFIG_USER_HELLO} += hello
...
```

5. Create the **Makefile** under *user/hello* directory as mentioned below to compile the hello application.

```
EXEC = hello
OBSJ = hello.o
all: $(EXEC)

$(EXEC): $(OBSJ)

$(CC) $(CFLAGS) $(LDFLAGS) -o $@ $(OBSJ) $(LDLIBS)

romfs:

$(ROMFSINST) /bin/$(EXEC)

clean:

-rm -f $(EXEC) *.elf *.gdb *.o
```

6. Build the kernel with this application and issue the following command to view the output of this application.

```
/>hello ↵
```

Here, you will see the output of your application.

5.7 Build New User Application Using SLS IP Drivers

To build a new user application using SLS IP drivers refer **DRIVERSAPI.txt** located at */Docs*.

5.8 Shell Commands

Using Shell commands you can perform operations on the uClinux on the NEEK board. Shell is the basic application on the Linux system, default shell provided in BSP is “sh”. “sh” uses the current directory as the prompting string. Commands can be executed under shell. (It works the same way as PC Linux). Key in help under shell will display the internal commands provided by shell.

Table below lists the shell commands useful while debugging:

| Command | Definition | Description |
|----------|---|--|
| cat | cat filename | Shows file on screen |
| cd | cd [directory] | Change current directory |
| chgrp | chgrp GROUP FILE... | Change the group membership of each FILE to GROUP |
| chmod | chmod mode file/dir | Change file/directory mode |
| chown | chown group:user file/dir | Change file/directory own |
| cmp | cmp file1 file2 | Compares two files |
| date | date [MMDDhhmm[YYYY]] | Get/set date |
| cp | cp file1 file2 | Copy source to destination |
| df | df [device] | Shows information about the filesystem on which each FILE resides, or all filesystems by default |
| echo | echo arguments [>filename] | Output the ARGs or redirect to file |
| exec | exec file | Exec FILE, replacing this shell with the specified program |
| exit | exit [N] | Exit the shell with a status of N. If N is omitted, the exit status is that of the last command executed |
| free | free | Shows memory status |
| help | help | Shows help message |
| hexdump | hexdump file | Hex dump file |
| hostname | hostname | Shows host name |
| Kill | kill [-s sigspec -n signum -sigspec] [pid job]...\n or kill -l [sigspec] | Sends signal to process |

| | | |
|----------|--------------------------|---|
| ln | ln -s file1 file2 | Creates a link to the specified TARGET |
| ls | ls [options] | List information about the FILES |
| mkdir | mkdir dirname | Creates the DIRECTORY |
| mknod | mknod type major minor | Creates device file |
| more | more filename | File perusal filter |
| mount | mount -t type device dir | Mount file system |
| mv | mv source dest | Rename SOURCE to DEST, or move SOURCE(S) to DIRECTORY |
| printenv | printenv | Print environment variables |
| pid | pid | Shows current process |
| ps | ps | Shows process information |
| pwd | pwd | Show current directory |
| quit | quit | Quit current process |
| rm | rm file | Remove file |
| rmdir | rmdir dir | Remove dir |
| sleep | sleep number | sleep several seconds |
| setenv | setenv var value | Set environment variable |
| source | source file | Run command in file |
| sync | System sync | sync |
| touch | touch [option] file | Update the access and modification times of each FILE to the current time |
| umask | umask octal number | The user file-creation mask is set to MODE |
| umount | umount dir | Umount file system |

6. Configuring Network utilities and NFS (Client)

This chapter introduces you about the network utilities ftp, dhcpd, telnet, boa, and inetd. First of all follow all steps for Ethernet driver mentioned in the section [SLS Ethernet IP Driver](#).

6.1 Configuring DHCP Client

1. Select **Customize Application/Library Settings**. Do the following settings for Network Application and Busybox.

```
Network Applications ---->
[*] dhcpd-new (2.0/2.4)
[*] ntpdate
[*] route
BusyBox ---->
[*] ifconfig
[*] Enable status reporting output (+7k)
[*] ping
```

2. To configure the IP address of the board dynamically (automatically), issue the following command on the target.

```
/>dhcpd ↵
```


3. To know the IP address received from the above command and to know the Ethernet port configuration, Issue following command on the target.

```
/>ifconfig ↵
```

You will find IP assigned to your board by ifconfig command

4. Ping your board from the host with the IP address (xx.xx.xx.xx) that you got from step 3.

```
#ping xx.xx.xx.xx ↵
```

 Make sure that DHCP server is available in your network and there is enough space for new IP allocation.

6.2 Static IP Allocation

To configure the Static IP, do the following settings:

```
BusyBox ---->
[*] ifconfig
[*] Enable status reporting output (+7k)
```

To allocate static IP to the board, issue following command on the target.

```
/>ifconfig eth0 192.168.0.14 ↵
```

6.3 Mounting NFS on NEEK

Before mounting NFS on NEEK, select the options as mentioned in [SLS Ethernet IP Driver](#) and [NFS](#). Then go to the user application and select the following.

```
Network Applications --->
[*] portmap
BusyBox --->
[*] BusyBox
Linux System Utilities --->
[*] mount
[*] Support mounting NFS file systems
Networking Utilities --->
[*] ifconfig
[*] Enable status reporting output (+7k)
```

Now build the kernel.

6.3.1 NFS Server (Host) Set Up

To set up the NFS server follow the steps below:

1. Login as a **super user** on your server on the host pc in your network.
2. Create a directory called **nfs** in the */home* directory.

```
#mkdir /home/nfs
```

3. Edit the file named **exports** under */etc* directory and add the following line:

```
/home/nfs 192.168.0.0/255.255.255.0(sync,no_root_squash,rw)
```



This setting may differ as per your network

5. Restart NFS server.

```
#service nfs restart
```

6. Verify it by issuing the following command.

```
#showmount -e
```

6.3.2 NFS Client (NEEK) Setup

When the **zImage** is up and running, issue the following command:

```
/>portmap &
```

```
/>mount -t nfs -n -o nolock,rsize=1024,wsiz=1024  
192.168.0.26:/home/nfs /mnt
```

After successful mounting, you can access the */home/nfs* directory on the NFS server (host PC) using */mnt* directory on the NFS client (NEEK board- target).

6.4 Configuring inetd, telnetd, ftpd Server

To configure inetd, telnetd, ftpd, follow the steps below:

1. First of all follow all steps for Ethernet driver mentioned in NEEK BSP driver chapter.

```
Network Applications ---->  
[*] ftpd  
[*] inetd  
[*] telnetd
```

2. When the kernel is up and running run write the following command on console:

```
/>inetd & ↵
```

This would run inetd server in the background which in turn will run the telnet and ftp server.

3. Now connect FTP client on HOST to the server running on the Board. Here Board IP is configured as 192.168.0.14 so issue:

```
#ftp 192.168.0.14 ↵
```

You will see following on the host terminal. Enter the ftp user name and password

```
Connected to 192.168.0.14
```

```
220- Welcome to the uClinux ftpd!
```

```
220 uClinux FTP server (GNU inetutils 1.4.1)ready
```

```
User (192.168.0.14 :( none)): ftp
```

```
331 Guest login ok, type your name as password.
```

```
Password: ↵
```

```
230 Guest login ok, access restrictions apply.
```

```
ftp>
```

4. Now connect Telnet client on HOST to the server running on the Board. Here Board IP is configured as 192.168.0.14 so issue:

```
#telnet 192.168.0.14 ↵
```

Target command shell will be opened.

6.5 Configuring boa Server

To configure the boa server on the board, follow the steps below:

1. Do the following selection.

```
[*] Customize Vendor/User Settings (NEW) ---->
    Network Applications ---->
        [*] boa
```

2. Issue following command on the target.

```
/>boa & ↵
```

3. On host PC, go to your web browser, search for **board_ip** in the address bar. uClinux default webpage opens.

7. Debugging Kernel and User Application

To debug the user application you will require following software and hardware:

- Cross/Straight Network cable
- JTAG cable
- Eclipse IDE for C/C++ developers. You can download Eclipse IDE from the

<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/20071103/eclipse-cpp-europa-fall2-linux-gtk.tar.gz>

7.1 Debugging uClinux kernel using nios2-elf-insight over JTAG

To debug the uClinux kernel using nios2-elf-insight over JTAG, first build the zImage by using the following steps:

Before debugging, configure the kernel as explained below:

```
#make menuconfig ↵
```

The menu configuration window opens:

```
Kernel hacking ---->
  [*] Kernel debugging
  [*] Compile the kernel with debug info
```

Save, exit and build the zImage.

Following are the steps to debug uClinux kernel using JTAG.

1. Download the .sof file to your board using command:

```
#nios2-configure-sof/home/uClinux/System-files/
sls_neek_bsp_hw.sof ↵
```

2. Go to **home/uClinux/uClinux-dist/linux 2.6.x** .You will find **vmlinux**. Copy it and paste it in same dir with name **vmlinux.elf**.
3. Now there will be debug script called **nios2-debug** in NIOS II IDE. The script is located at *<altera_tools_installation_path>/nios2eds/bin*.
4. Modify the script to make it compatible with uClinux kernel. Save a copy of this script before modifying it and change line no 70 in nios2-debug from :

```
break *main

to

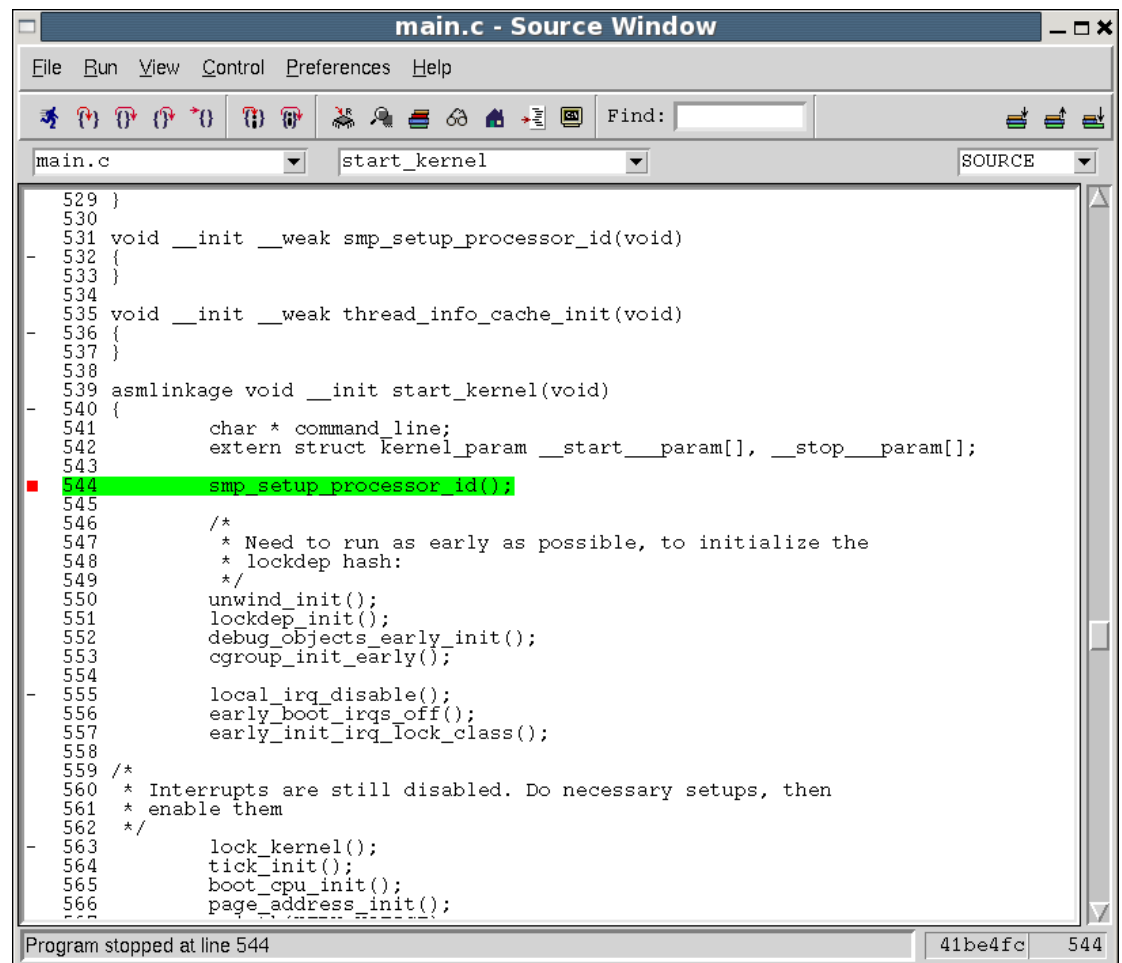
break *start_kernel
```

5. Issue the command:

```
#nios2-debug home/uClinux/uClinux-dist/linux
2.6.x/vmlinux.elf ←
```

Wait for few seconds and following window will get opened.

Figure 17 : main.C



Now you can debug easily with aid of GUI. You can set the break point, check the content of registers and see the assembly code also.

- When you want to perform kernel debugging then only use the modified script otherwise use the default script.

7.2 Debugging User Application using gdbserver over Ethernet

Insight is the GUI of gdb debugger. Often it's not easy to remember the debug Command. Therefore insight provides the graphical interface so user can debug easily. The aim of the document is to explain how to debug user space application. Steps are mentioned below:

For debugging user application over Ethernet you should have proper zImage .To build zImage, Follow **Driver for Ethernet in NEEK BSP DRIVERS**.

Here is the example of simple helloworld.c program debugging:

1. Select gdbserver (old) in menuconfig under user application → miscellaneous application.
2. Select the ifconfig utility in your busybox.
3. On your host set PATH environmental variable to toolchain with following command:

```
#PATH=$PATH:/home/uClinux/Bintools/opt/nios2/bin ↵
#export PATH ↵
```

4. Compile the application with debug symbols and no optimization, on your host, Here is helloworld.c example. Your current working dir should have this .c file.

```
#nios2-linux-uclibc-gcc -g helloworld.c -o helloworld
-elf2flt ↵
```

This will generate **Helloworld** and **Helloworld.gdb** files in your current working dir. Copy generated Helloworld to **romfs/bin** in uClinux-dist dir.

5. Build the image, download it to target and configure the Ethernet IP, 192.168.0.14 through following command

```
/>ifconfig eth0 192.168.0.14 ↵
```

Then start the gdbserver listening on an unused port eg. 9999, with command:

6. Apply the command given below:

```
/> gdbserver localhost:9999 /bin/Helloworld ↵
```

7. It will display,

```
Process /bin/Helloworld created; pid = 20
Listening on port 9999
```

8. Next, on your Linux PC, in the hello source dir, run insight gdb,

```
#nios2-linux-uclibc-insight Helloworld.gdb ↵
```

9. A source window will open and display the source Helloworld.c.
10. The open a gdb console, with **View>Console**, enter gdb command in this window.

```
gdb>target remote 192.168.0.14:9999 ↵ (board_ip:9999)
```

11. Then it will report the target address of the program. Now you can set break points and debug. On target you can see the message: Remote debugging from host IP, here you will see your IP (IP of host) address.

```
gdb> b main ↵ (insert break point at main)
```

```
gdb> c ↵ (continue)
```

12. Now you can debug via command or through GUI of insight. Insight GUI are quite easy to understand. If you want to use command then you can use:

```
gdb> s ↵ (for single stepping)
```

```
gdb> r ↵ (to run program )
```

7.3 Debugging User application using Eclipse IDE

To debug the user application, follow the steps below.

1. Follow all steps mentioned in Ethernet driver selection, from chapter [Configuring Device Driver](#).
2. Build the uClinux kernel by selecting

```
User application ---->
    miscellaneous application ---->
        [*] gdbserver(old).
```

3. Open the terminal. Change directory to the Eclipse IDE folder
4. Write the command on the terminal to open the Eclipse IDE.

```
#./eclipse ↵
```

5. Type **workspace** to choose your workspace.
6. Select **File>New>C project** to Create a new project.
7. Enter **Helloworld** as your Project name.

8. Project types, executable, Hello world ANSI C Project, (or empty project and add your source).Finish.
9. Now, setup for nios2 tool chain in Eclipse IDE on the host by following the steps below:
 - a) Select the project **Helloworld** under **Project Explorer**.
 - b) Right click on **Helloworld** and select **Properties**. Project properties dialog box opens.
 - Select **Settings>GCC C Compiler** under **C/C++ build**. Make following settings:
 - Command: **nios2-linux-uclibc-gcc**
 - All options: **-O0 -g -Wall -c -fmessage-length=0**
 - Click **Apply**.
 - Select **Settings>GCC C Linker** under **C/C++ build** and make following settings
 - Command: **nios2-linux-uclibc-gcc**
 - All Options: **-elf2flt**
 - Select **Miscellaneous** under **GCC C Linker**
 - Linker flags: **-elf2flt**
 - Click **Apply**
 - c) Select **Run/Debug Settings** under Project properties. Run/Debug settings dialog box opens
 - Click on **New** button. Select Configuration Type dialog box opens
 - Select C/C++ Local application.
 - Click **OK**. Properties for new Configuration dialog box opens.
 - Under **Main** tab do the following settings:
 - Browse the **Helloworld** project under project.
 - Type **Debug/Helloworld.gdb**
 - Select Debugger tab. Make following settings
 - Debugger: **gdbserver Debugger**
 - Under **Main** tab of Debugger options make following settings
 - GDB Debugger: **nios2-linux-uclibc-gdb**
 - Under **Connection** tab
 - Type: **TCP**
 - Host name or IP address: **192.168.0.14 (IP address of NEEK board)**
 - Port Number: **9999** (you can give any number)
 - Click **Apply** and click **OK**. You have finished setting of run/debug settings.
 - Click **Apply** and click **OK**. This finishes project properties settings.
 - d) Select **Project>Build Configuration>Build>All** to build your project.

- Select workspace>Helloworld>Debug>Helloworld and copy the file into /uclinux-dist/romfs/bin.
- e) Build the zImage as explained in early section.

f) Click **OK**.

10. Now, select **Project-->Build project** to compile your project.

11. Copy from your **workspace/Helloworld/Debug/Helloworld** to **romfs/bin** in uClinux-dist and build the zImage as explained earlier.

12. Configure the **SOF** on your board and download the **zImage**.

13. The kernel starts booting. After successful booting, configure the IP address of the board (same as entered in the Eclipse IDE). Here IP is: 192.168.0.14

```
/>ifconfig eth0 192.168.0.14 ↵
```

14. Now issue the command:

```
/>gdbserver localhost:9999 /bin/Helloworld ↵
```

15. It will display:

```
Process /bin/Helloworld created; PID = 20
Listening on port 9999
```

16. Now move to the IDE on host.

a) Select **Run>Open Debug Dialog**

- Select **Helloworld** as Debug application
 - C/C++ Application: **Debug/Helloworld.gdb**
- Click **Apply**.
- Click **Debug**.

b) If there is no error then debug perspective opens and it will establish a connection with the board. You will see the following message on the target.

```
remote debugging from: 192.168.0.26 (your host ip)
```

17. Now you can debug easily with the help of step into/step over feature of IDE.

8. Demonstrations & Quick Reference

Download **Demonstrations and Quick Reference** from <http://slscorp.com/pages/bspdownload.php>. This demo package contains two types of applications:

- Applications supported to NEEK Application Selector
 - SLS uClinux demo
 - SLS_Player

These demonstrations are not the final product but are for demo purpose only. It demonstrates how to develop real time application using the uClinux and nios2.

8.1 Demonstrations

8.1.1 NEEK Application Selector Demos

NEEK Application Selector demos are developed using NEEK BSP. The demos are located at */Demos/NEEK_Application_Selector_Demos* directory. To run the demos, the board should be preconfigured with Altera application selector utility.

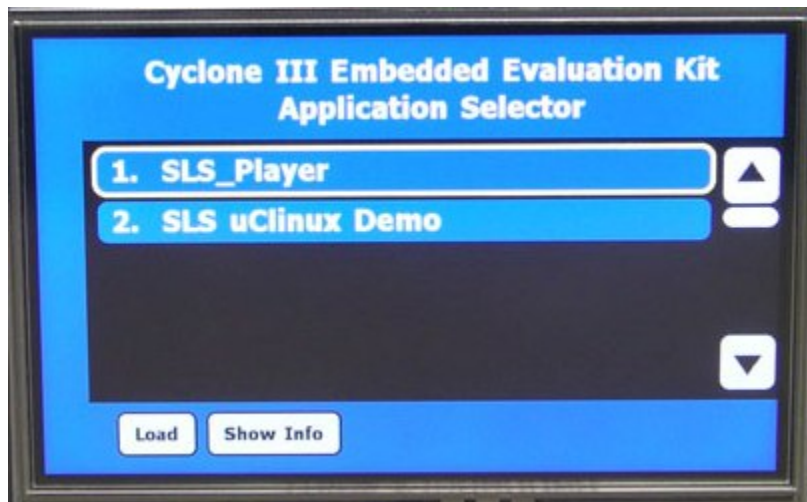
SLS_Player

It is a media player application designed by SLS to play the MP3 files and MPEG 1/2 files without audio information.

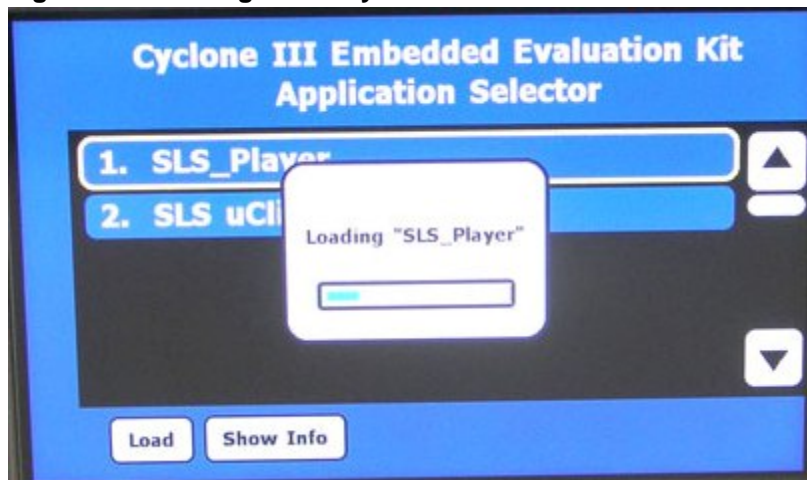
To know how to run this demo, refer **HowToRun.txt** located at *Demo/NEEK_Application_Selector_Demos/Media_Player*.

Follow the steps mentioned below to run the SLS uClinux demo.

1. Power on the board by pressing the switch SW1. You will see the Application Selector menu on the LCD Touch Screen Display. See [Figure 18](#)

Figure 18 : Application Selector

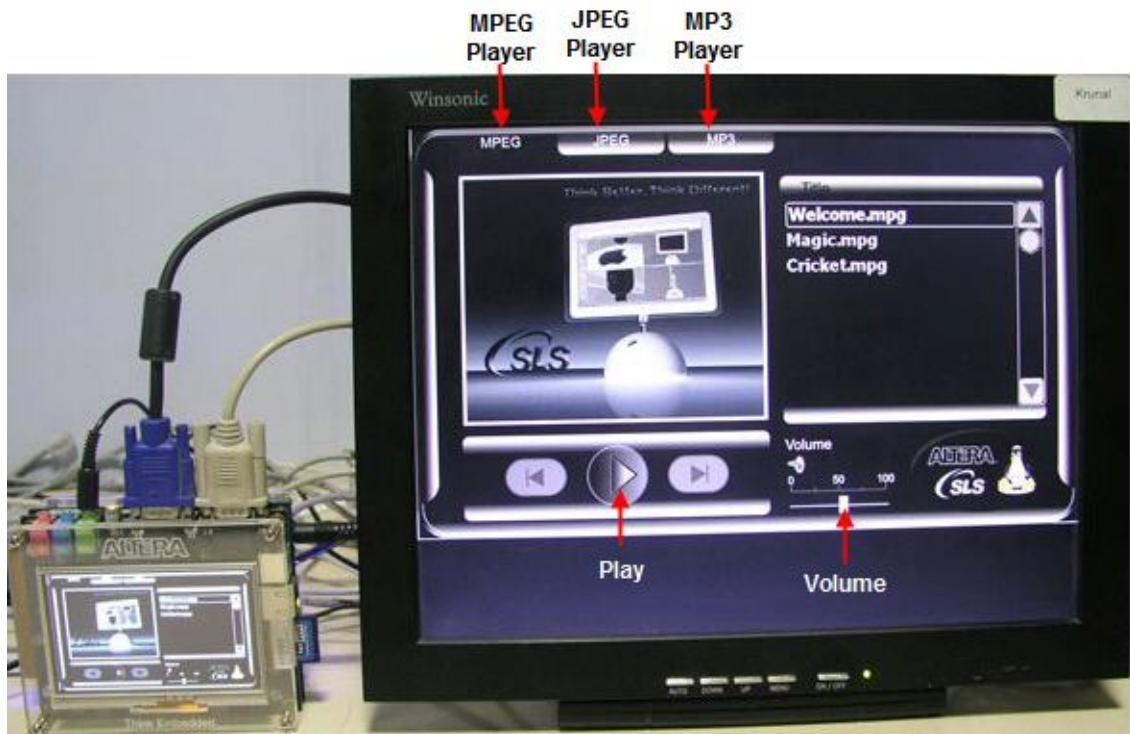
2. In the application selector main menu, touch the SLS_Player to select it.
3. Touch the Load button located on the bottom left corner of the Touch Screen to load the SLS Player application. You will see [Figure 19](#)

Figure 19 : Loading SLS Player

4. The SLS Player information dialog box displays as shown in [Figure 20](#) below.

Figure 20 : Running the SLS Player - Main Menu

5. Change the SD Card mode as mentioned in dialog box and click **OK**. The SLS Player main menu displays.
6. Select MPEG to play MPEG clips. See [Figure 21](#) below.

Figure 21 : SLS Player Main Menu

7. Select **Welcome.mpg** and click **Play**. The MPEG clip will get played on the screen.
8. Repeat steps 6 to 7 to play JPEG and MP3 files

SLS uClinux Demo

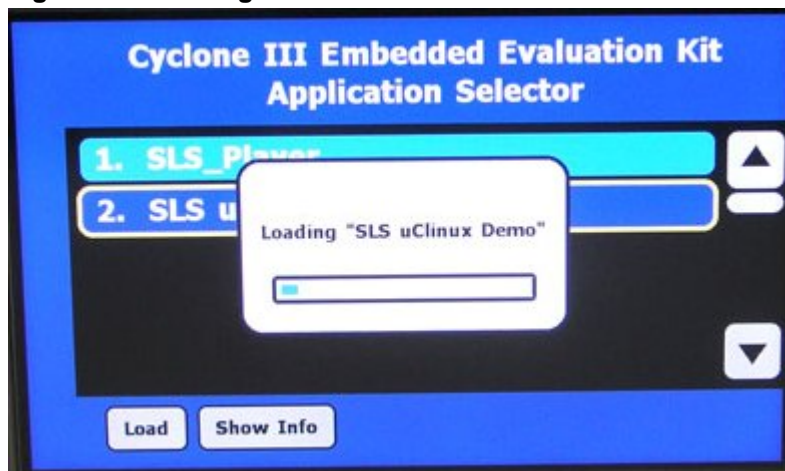
The SLS uClinux demo is designed to run multiple applications on uClinux platform. The application uses SLS GUI library and various SLS IP cores including Ethernet Mac 10/100, TPC and VGA /LCD Controller.

To know how to run this demo, refer **HowToRun.txt** located at *Demo/NEEK_Application_Selector_Demos/Multiple_Processing*.

Follow the steps mentioned below to run the SLS uClinux demo.

1. Power on the board by pressing the switch SW1. You will see the Application selector menu on the LCD Touch Screen Display.
2. In the application selector main menu, touch the SLS uClinux Demo to select it.
3. Touch the Load button located on the bottom left corner of the Touch Screen to load the SLS uClinux demo application. You will see the [Figure 22](#)

Figure 22 : Loading SLS uClinux Demo

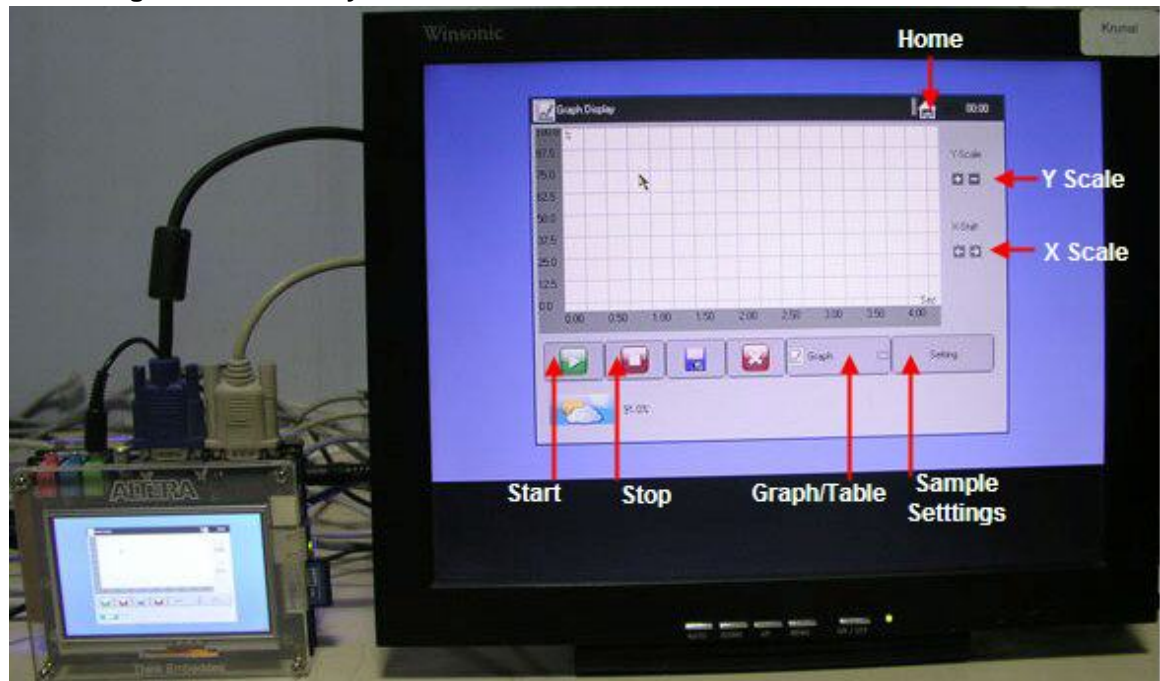


4. The SLS uClinux Main menu displays as shown in [Figure 23](#) below.

Figure 23 : Running the SLS uClinux Demo - Main Menu

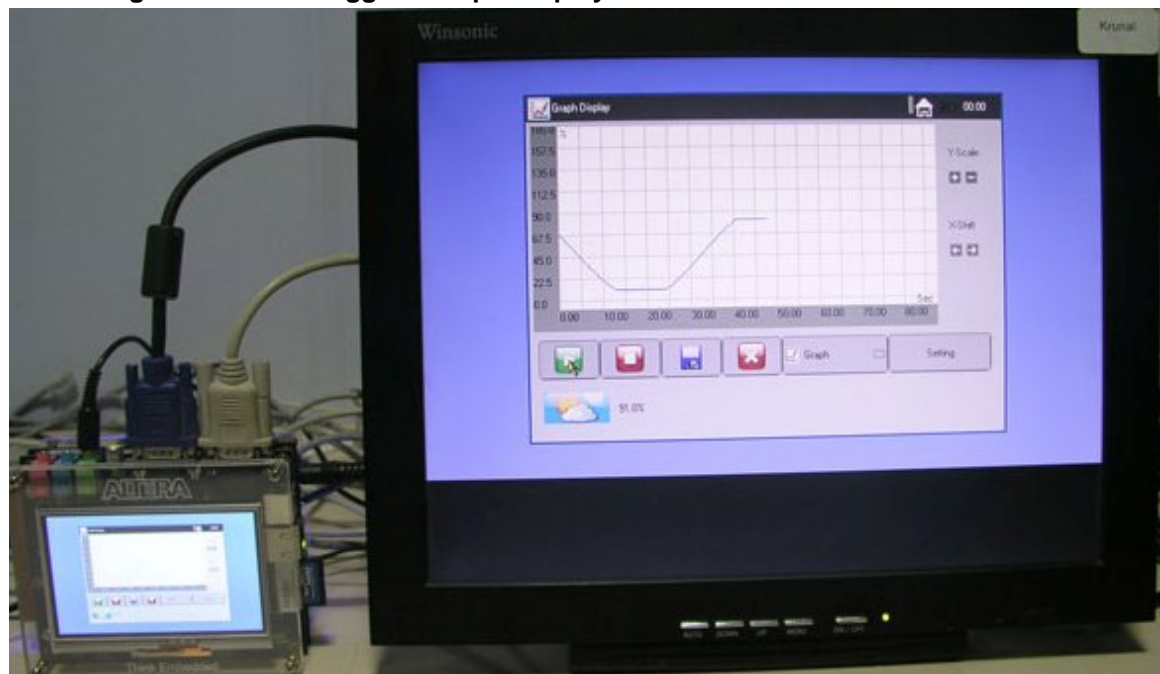
5. Select **Start Experiment** option. You will see a blank graph window as shown in [Figure 24](#).

Figure 24 : SLS Player Main Menu



6. Select **Start** button to start logging the dummy data. The graph will be drawn as shown in [Figure 25](#) below.

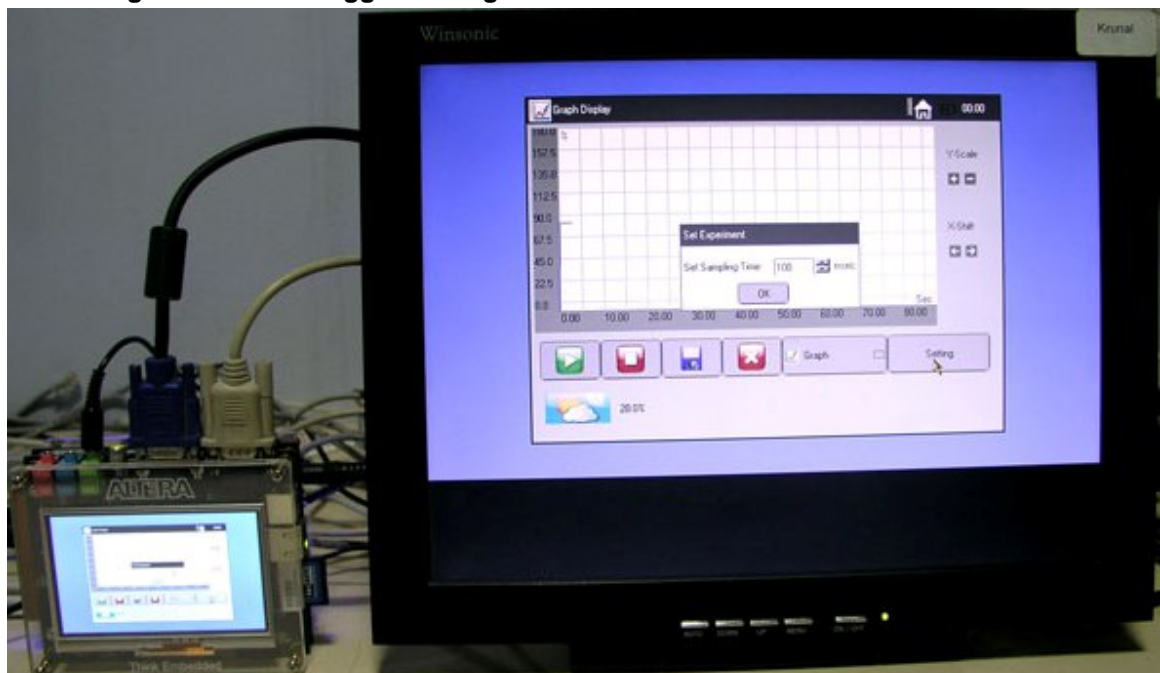
Figure 25 : Data Logger - Graph Display



7. To view the logged data in table view, select **Table** option. See [Figure 26](#) below.

Figure 26 : Logged Data in Table View

8. To change sampling frequency click on Settings button. Enter the desired frequency. See [Figure 27](#) below.

Figure 27 : Data Logger Settings

8.2 Quick Reference

Quick Reference gives you quick overview of the applications explained in chapter 5 and 6. When you extract Quick Reference, you will find following directories:

- Prebuilt_zImage
- Framebuffer_console

8.2.1 Prebuilt_zImage

Prebuilt_zImage directory contains files to format the SD Card and other ready to use applications:

- zImage
- sls_neek_bsp_hw.sof
- readme.txt

Before going further, please refer the readme.txt.

SD Card formatting for Application Selector

Download .sof and zImage. You will get console on serial UART. When the zImage is up and running, plug the SD card into the socket (NEEK board). Issue the following command to format the SD card.

```
/>mkdosfs -I -F 16 /dev/hda
```

Other Applications

zImage given here is a ready reference to test all applications explained in chapter 5 and 6. See [User Applications](#) and [Configuring Network utilities and NFS \(Client\)](#) for more details. Using this zImage you can only explore some of the capabilities of BSP.

8.2.2 Framebuffer_console

Framebuffer_console directory contains files to boot the uClinux to get the console on frame buffer (NEEK LCD).

- zImage
- sls_neek_bsp_hw.sof
- readme.txt

Before going further, please refer the readme.txt.

Booting uClinux on NEEK LCD

Download .sof and zImage, you will see booting uClinux on NEEK LCD. When the zImage is already up and running, plug PS/2 keyboard in PS/2 port. Now execute the shell command as usual. You can execute all shell commands offered by shell and can run various programs and utilities available in the zImage.

After successfully running the zImage, you will see the booting of uClinux on the NEEK LCD. See [Figure 28](#) and [Figure 29](#).

Figure 28 : Framebuffer Console View 1



Figure 29 : Framebuffer Console View 2

